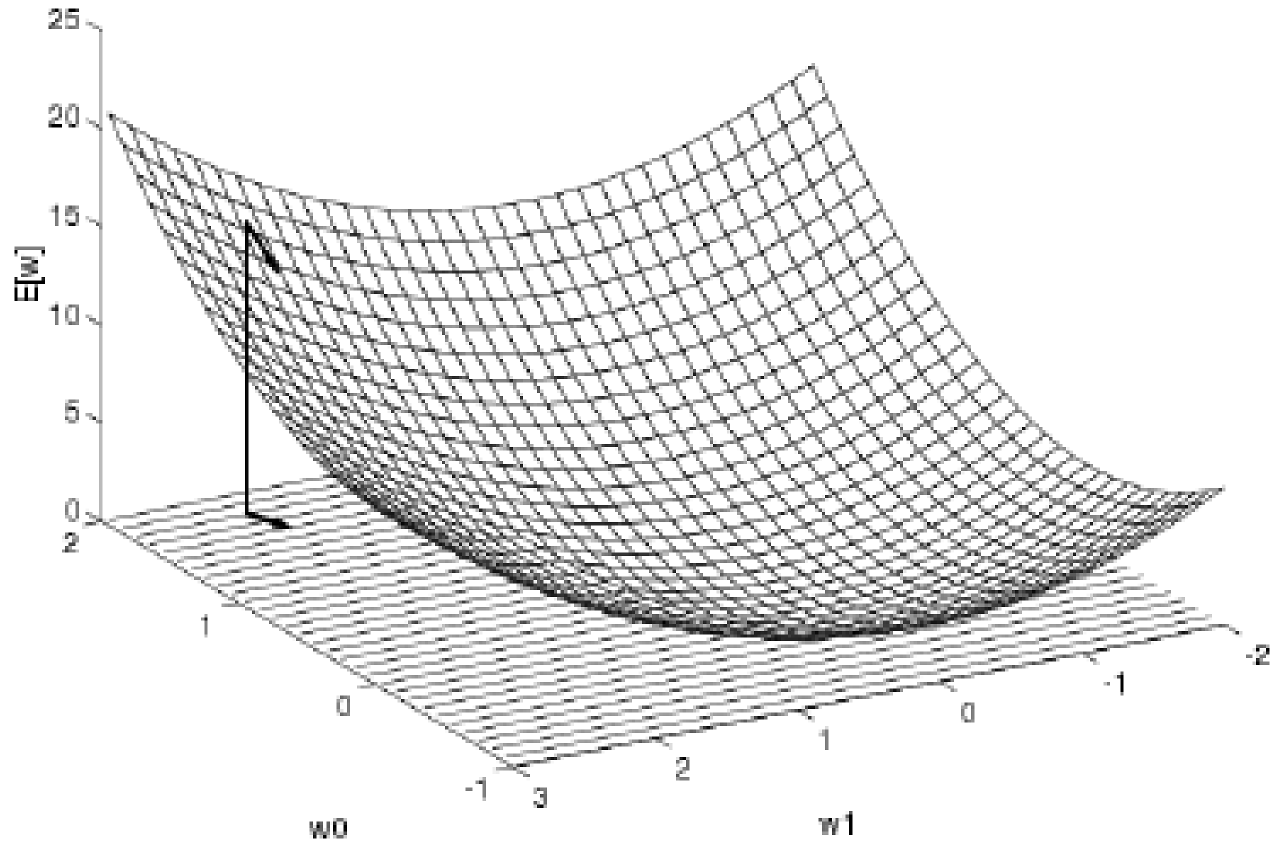
Machine Learning

CSE 343/543

Lecture 4

Maximum Likelihood and Linear Models for Regression



Gradient vector points to the direction of fastest increase of the function;
So we take the –ve gradient to move in the direction of steepest descent.

Regression

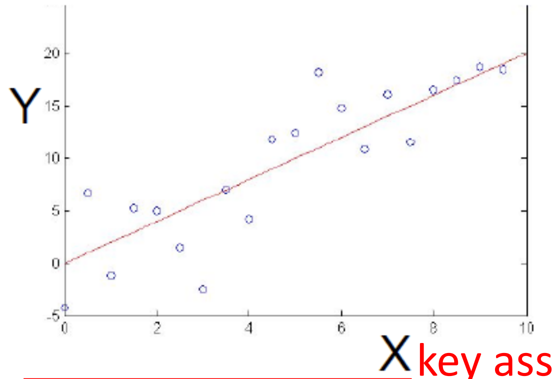
Wish to learn $f: X \rightarrow Y$, where Y is real, given $\{<x^1,y^1>\ldots<x^n,y^n>\}$

Approach:

1. choose some parameterized form for $P(Y|X;w)$
   (w is the vector of parameters)

2. derive learning algorithm as MLE or MAP estimate for w

1. Choose parameterized form for P(Y|X; θ)



key assumption

Assume Y is some $\boxed{\text{deterministic } f(X)}$, plus random noise

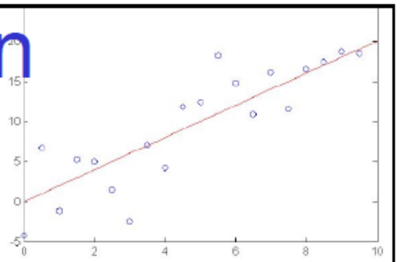
$$y = f(x) + \epsilon \qquad \text{where} \quad \epsilon \sim N(0, \sigma^2)$$

Therefore Y is a random variable that follows the distribution

$$p(y|x) = N(f(x), \sigma^2)$$

and the expected value of y for any given x is $E_{p(x,y)}[y]=f(x)$

Consider Linear Regression



$$p(y|x) = N(f(x), \sigma^2)$$

E.g., assume f(x) is linear function

$$f(x) = w_0 + \sum w_i x_i$$

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma^2)$$

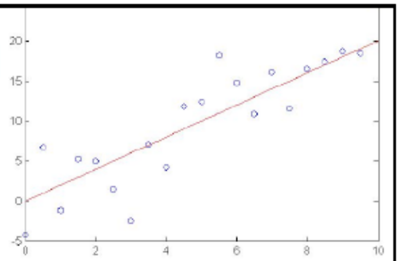
$$E_{p(x,y)}[y|x] = w_0 + \sum_i w_i x_i$$

Notation: to make our parameters explicit, let's write

$$W = < w_0, w_1 \ldots w_n >$$

$$p(y|x; W) = N(w_0 + \sum_i w_i x_i, \sigma^2)$$

Training Linear Regression



$$p(y|x;W) = N(w_0 + w_1 x, \sigma^2)$$

How can we learn W from the training data?

Learn Maximum Conditional Likelihood Estimate!

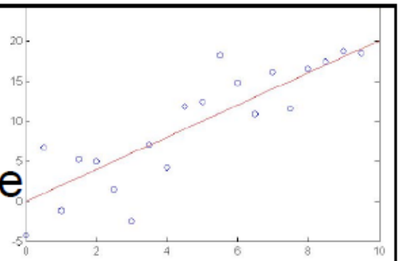
$$W_{MCLE} = \arg\max_W \prod_l p(y^l|x^l, W)$$

$$W_{MCLE} = \arg\max_W \sum_l \ln p(y^l|x^l, W)$$

where

$$p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}\left(\frac{y-f(x;W)}{\sigma}\right)^2}$$

Training and Regression



Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\max_W \sum_l \ln p(y^l | x^l, W)$$

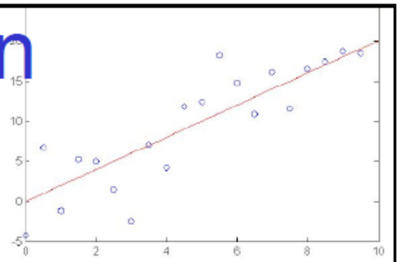
where $$p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$

$$\sum_l \ln p(y^l | x^l; W) =$$

so: $$W_{MCLE} = \arg\max_W \sum_l -(y^l - f(x^l;W))^2$$

$$= \arg\min_W \sum_l (y^l - f(x^l;W))^2$$

Consider Linear Regression



$$p(y|x) = N(f(x), \sigma^2)$$

E.g., assume f(x) is linear function of x

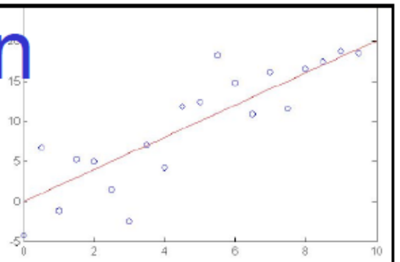
$$f(x) = w_0 + \sum_i w_i x_i$$

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma^2)$$

Can we derive gradient descent rule for training?

$$\frac{\partial \sum_l (y - f(x;W))^2}{\partial w_i} = \sum_l 2(y - f(x;W)) \frac{\partial (y - f(x;W))}{\partial w_i}$$

Consider Linear Regression



$$p(y|x) = N(f(x), \sigma^2)$$

E.g., assume f(x) is linear function of x

Can we derive gradient descent rule for training?

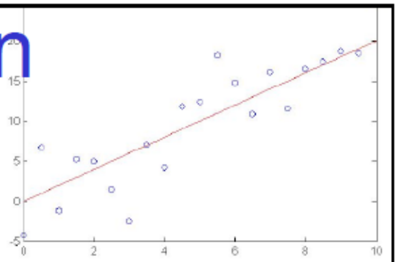
$$\frac{\partial \sum_l (y - f(x;W))^2}{\partial w_i} = \sum_l 2(y - f(x;W)) \frac{\partial (y - f(x;W))}{\partial w_i}$$

$$= \sum_l -2(y - f(x;W)) \frac{\partial f(x;W)}{\partial w_i}$$

And if $f(x) = w_0 + \sum_i w_i x_i$ ...

Gradient descent rule: $\boxed{w_i \leftarrow w_i + \eta \sum_l (y^l - f(x^l;W)) \; x_i^l}$

Consider Linear Regression



$$p(y|x) = N(f(x), \sigma^2)$$

E.g., assume f(x) is linear function of $\boxed{\phi_i(x)}$

$$f(x) = \sum_i w_i \phi_i(x)$$

$$p(y|x) = N\left(\sum_i w_i \phi_i(x), \sigma^2\right)$$

$$\boxed{w_i \leftarrow w_i \qquad\qquad + \eta \sum_l (y^l - f(x^l; W))\ \phi_i(x^l)}$$

How about MAP instead of MLE estimate?

Let's assume Gaussian prior: each $w_i \sim N(0, \sigma^2)$

$$p(w_i) = \frac{1}{Z} \exp\left(-\frac{(w_i - 0)^2}{2\sigma^2}\right)$$

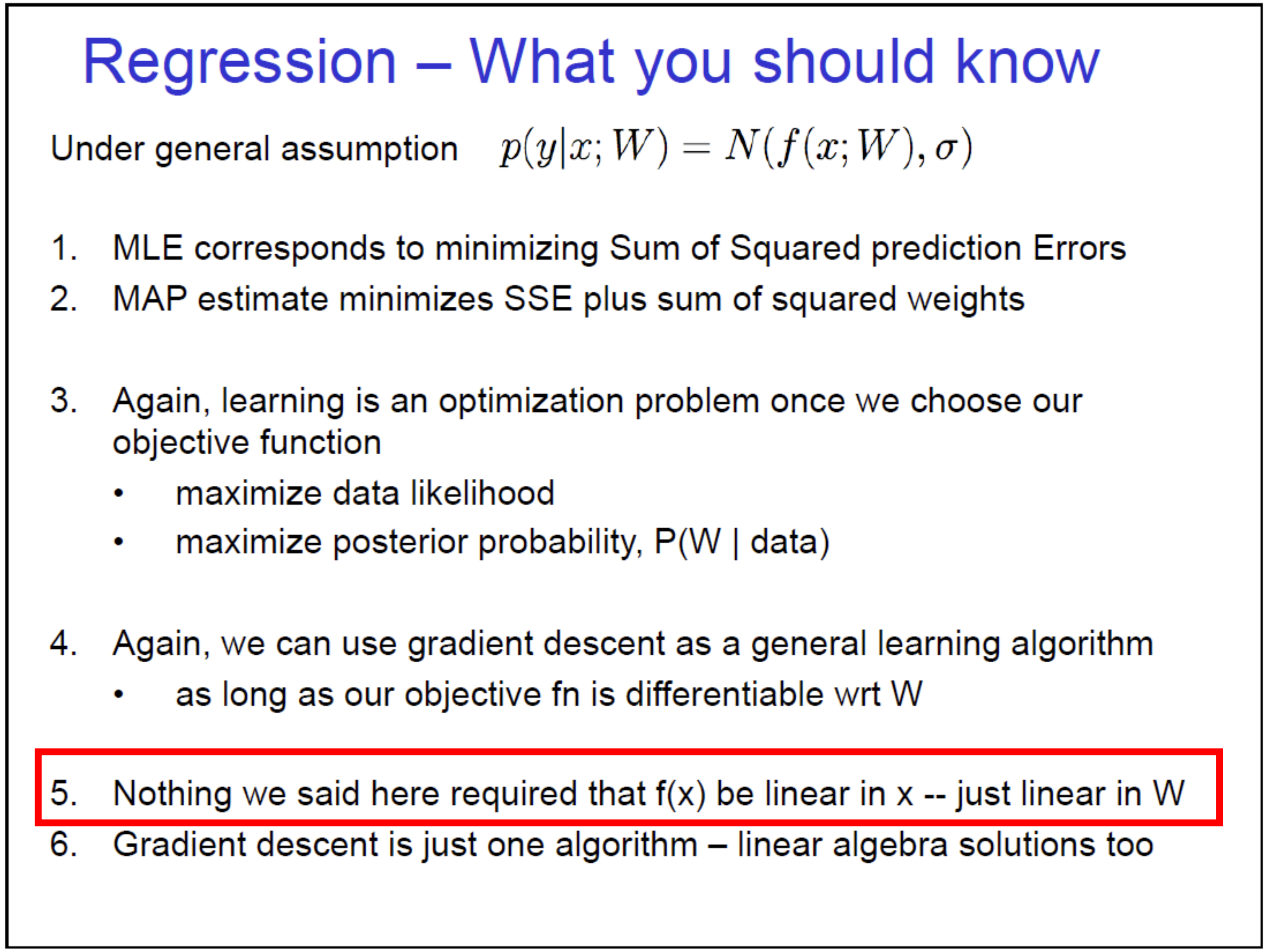
Then MAP estimate is

$$W = \arg\max_W -\frac{1}{2\sigma^2} \sum_{w_i \in W} w_i^2 + \sum_{l \in training\ data} \ln P(Y^l | X^l; W)$$

$$= \arg\min_W \frac{1}{2\sigma^2} \sum_{w_i \in W} w_i^2 + \sum_{l \in training\ data} (y^l - f(x^l; W))^2$$

Gradient descent:

$$\boxed{w_i \leftarrow w_i - \lambda w_i + \eta \sum_l (y^l - f(x^l; W))\ x_i^l}$$



Regression – What you should know

Under general assumption $p(y|x;W) = N(f(x;W),\sigma)$

1. MLE corresponds to minimizing Sum of Squared prediction Errors
2. MAP estimate minimizes SSE plus sum of squared weights
3. Again, learning is an optimization problem once we choose our objective function
   - maximize data likelihood
   - maximize posterior probability, P(W | data)
4. Again, we can use gradient descent as a general learning algorithm
   - as long as our objective fn is differentiable wrt W
5. Nothing we said here required that f(x) be linear in x -- just linear in W
6. Gradient descent is just one algorithm – linear algebra solutions too

Important to remember!!

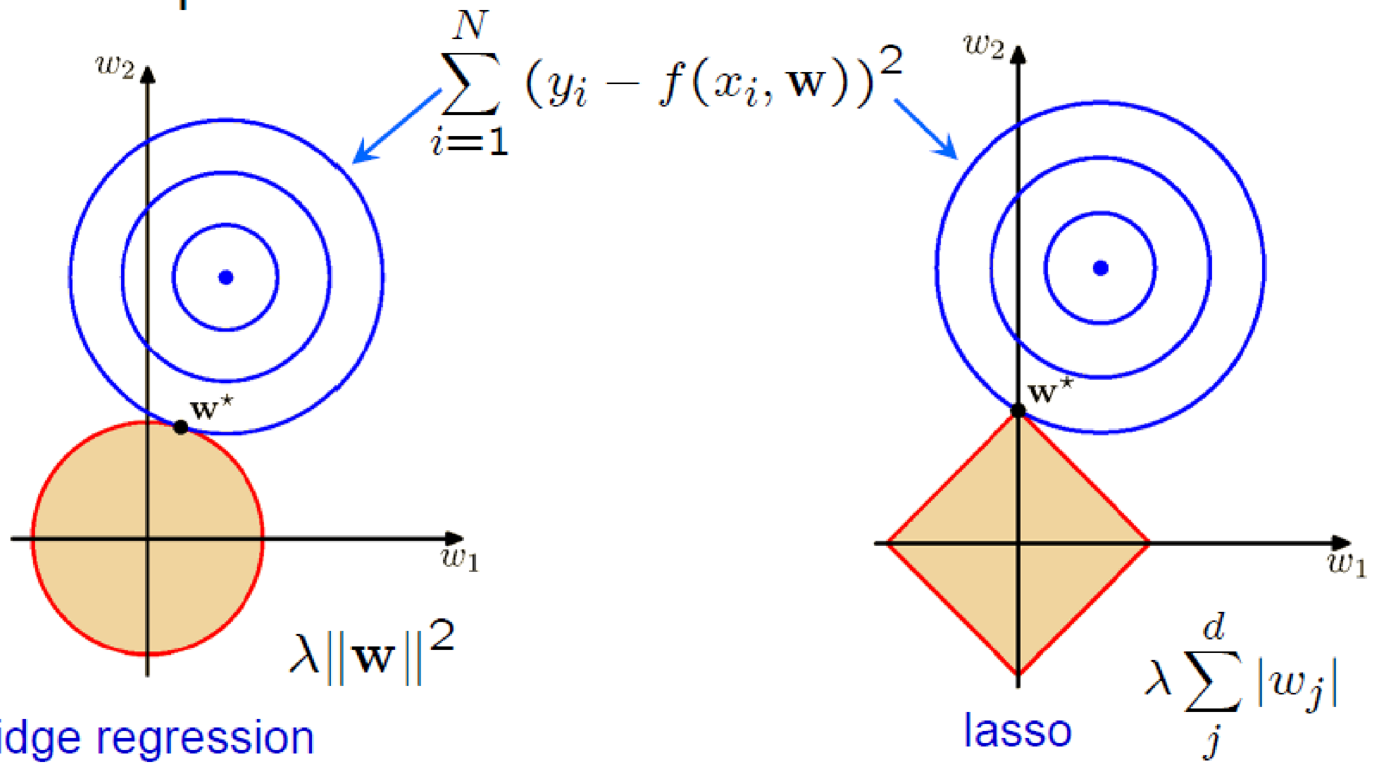
Regularization in Linear Regression

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \underbrace{\sum_{\ell} \left(y^{\ell} - f(\boldsymbol{x}^{\ell}, \boldsymbol{w})\right)^2}_{\substack{L(y, f(\boldsymbol{x}, \boldsymbol{w})) \\ \text{Loss function}}} + \lambda \underbrace{||\boldsymbol{w}||_2^2}_{\substack{R(\boldsymbol{w}) \\ \text{regularization}}}$$

- Regularization
  - Least squares regression – $R(\boldsymbol{w}) = 0$
  - Ridge regression – $R(\boldsymbol{w}) = ||\boldsymbol{w}||_2^2$
  - Least Absolute Shrinkage and Selection (LASSO) – $R(\boldsymbol{w}) = ||\boldsymbol{w}||_1$
- In general, p-norm regularization is $R(\boldsymbol{w}) = ||\boldsymbol{w}||_p = \left(\sum_{j=1}^{d} |w_j|^p\right)^{\frac{1}{p}}$

Effect of Regularization

- contour plots for d = 2



- Minimum where loss contours tangent to regularizer's
- For the lasso case, minima occur at "corners"
- Consequently one of the weights is zero
- In high dimensions many weights can be zero

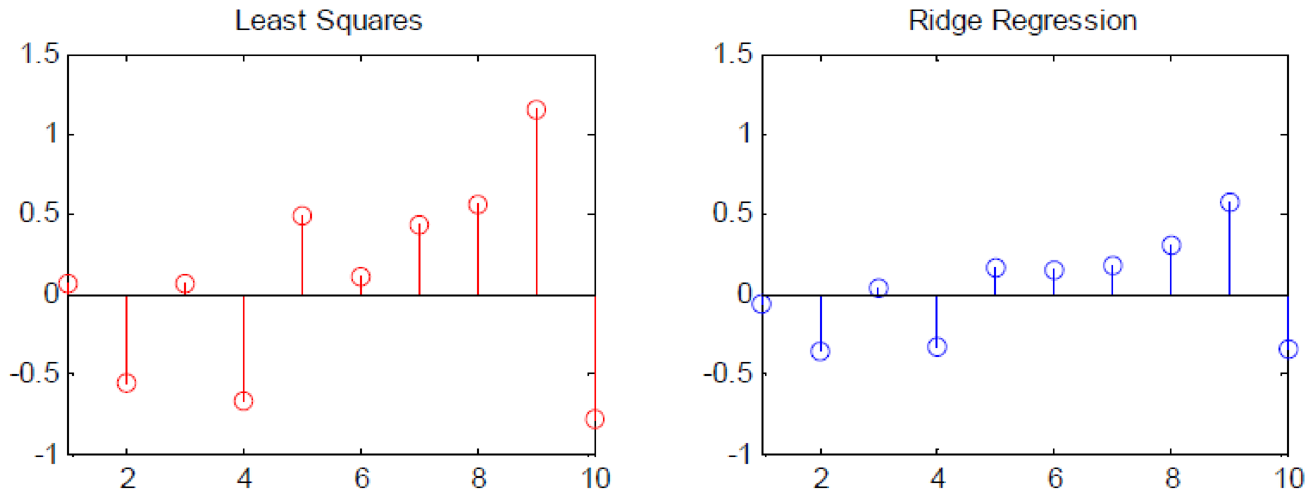
Comparison of learnt weight vectors

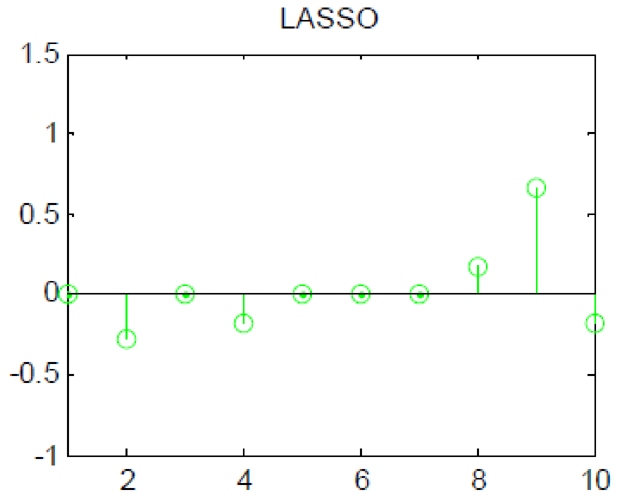
Linear regression

N = 100

d = 10

y = w . x



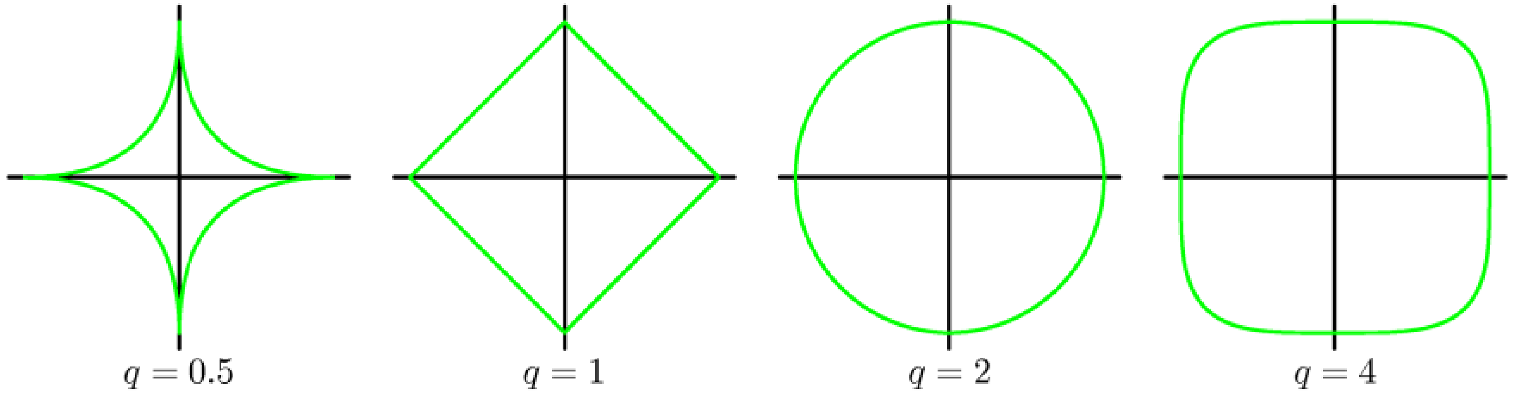


lambda = 1

Lasso: number of non-zero coefficients = 5

Other regularizers

$$\sum_{i=1}^{N} (y_i - f(x_i, \mathbf{w}))^2 + \lambda \sum_{j}^{d} |w_j|^q$$



- For $q \geq 1$, the cost function is convex and has a unique minimum. The solution can be obtained by quadratic optimization.
- For $q < 1$, the problem is not convex, and obtaining the global minimum is more difficult

Summary: Linear Regression

- Probabilistic Model: $y = \mathcal{N}\left(f(\boldsymbol{x}_i, \boldsymbol{w}), \sigma^2\right)$
- Parameters – ***w***

$$\mathcal{L}(y, f(\boldsymbol{x}, \boldsymbol{w})) = \sum_{i=1}^{N} \ell\left(f(\mathbf{x}_i, \boldsymbol{w}), y_i\right) + \lambda R(\boldsymbol{w})$$

$$\boxed{\widehat{y}_i = f(\boldsymbol{x}_i, \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}_i}$$

$$\widehat{y}_i = f(\boldsymbol{x}_i, \boldsymbol{w}) = w^0 x_i^0 + w^1 x_i^1 + w^2 x_i^2 + \cdots$$

x_i^j : j^{th} element of the i^{th} sample vector $\boldsymbol{x}_i$
Usually, $x_i^0 = 1$

- Squared loss

$$\ell\left(f(\mathbf{x}_i, \boldsymbol{w}), y_i\right) = \left(y_i - f(\mathbf{x}_i, \boldsymbol{w})\right)^2$$

Summary: Linear Regression

- Probabilistic Model: $\boldsymbol{y} = \mathcal{N}\left(f(\boldsymbol{x}_i, \boldsymbol{W}), \boldsymbol{\Sigma}\right)$ covariance
- Parameters – $\boldsymbol{W}$

$$\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{W})) = \sum_{i=1}^{N} \ell\left(f(\mathbf{x}_i, \boldsymbol{W}), \boldsymbol{y}_i\right) + \lambda R(\boldsymbol{W})$$

$$\boxed{\widehat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x}_i}$$

- Squared loss

$$\ell\left(f(\mathbf{x}_i, \boldsymbol{W}), \boldsymbol{y}_i\right) = ||\boldsymbol{y}_i - f(\mathbf{x}_i, \boldsymbol{W})||^2$$

Logistic Regression

Logistic Regression

- Basic Idea:
  - Logistic regression is the type of regression we use for a binary (or discrete) response variable ($Y \in \{0,1\}$)
  - Linear regression is the type of regression we use for a continuous, normally distributed response ($Y \in R^m$) variable
- Use a function to map real numbers to {0,1}
  - Thresholding
  - Some form of ‘squishing’ operation

Probabilistic Models

- Instead of modelling y_i as a Gaussian r.v. (as in Linear Regression),
  - model y_i as a Bernoulli random variable.
  - or more generally, as a binomial random variable.

Bernoulli and Binomial Distributions

- Bernoulli – one trial for a binary experiment

$$Pr(y|x;p) = \begin{cases} p, & y = 1 \\ 1-p, & y = 0 \end{cases}$$
$$= p^y (1-p)^{(1-y)}$$

- Binomial – n trials of the same binary experiments
  - Bernoulli is a special case of Binomial with n=1

$$\boxed{Pr(Y = y|x;n,p) = \binom{n}{y} p^y (1-p)^{n-y}}$$

Why model like this?
when x_i are discrete/mixed/grouped

- Example: x = age group of a person; y = voted/not voted
  - many individuals may get mapped to the same x

Why can't we use Linear Regression to model binary responses?

- The response (Y) is NOT normally distributed
- The variability of Y is NOT constant
  - Variance of Y depends on the expected value of Y
  - For a Y~Binomial(n,p) we have Var(Y)=np(1-p) which depends on the expected response, E(Y)=np
- The model must produce predicted/fitted probabilities that are between 0 and 1
  - Linear regression produce fitted responses that vary from -∞ to ∞

Recall that Linear Regression assumes a Normal distribution with fixed variance

Linear vs. Logistic Regression

- Linear Regression models try to predict the mean of y, i.e., f(x;w)

- Logistic Regression models try to predict ‘p’, the probability of a trial resulting in a success (Y=1)
  - p ϵ [0,1] (better than {0,1})
  - What if we try to model some invertible function of p?

- What function of ‘p’ should we model?

Probability and Odds of an Event

$$\text{odds}(Y=1) = \frac{P(Y=1)}{P(Y=0)} = \frac{P(Y=1)}{1-P(Y=1)}$$

$$= \frac{p}{1\text{-}p}$$

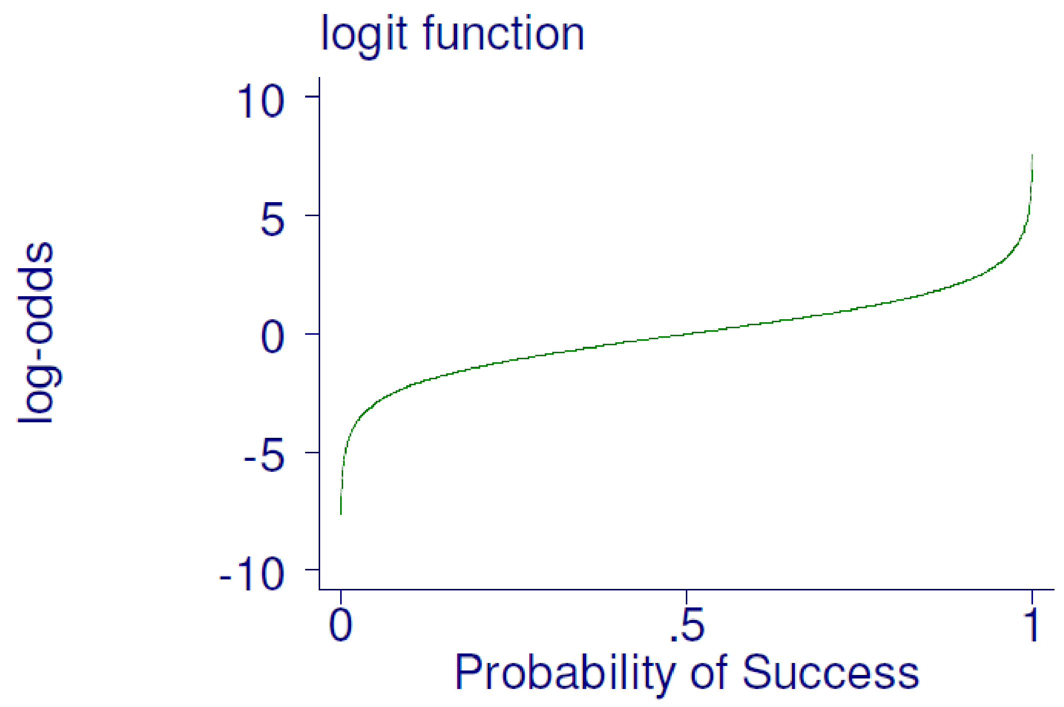
- We can go back and forth between odds and probabilities:

$$\text{Odds} = \frac{p}{1\text{-}p}$$

$$p = \text{odds}/(\text{odds}+1)$$

Logit Function (or log(odds))

- Why log(odds)?
  - Odds map ‘p’ from [0,1] to [0,∞]
  - Linear Regression predicts in the range [-∞ , +∞]
  - log(odds) maps odds from [0,∞] to [-∞ , +∞]
  - So we can use linear regression to predict log(odds)



Logistic Regression

- Same as Linear Regression on ‘log(odds)’

$$\log(\text{odds}) = w_0 + w^1 x^1 + w^2 x^2 + \cdots + w^d x^d = \boldsymbol{w}^\top \boldsymbol{x}$$

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = w_0 + w^1 x^1 + w^2 x^2 + \cdots + w^d x^d = \boldsymbol{w}^\top \boldsymbol{x}$$

$$\left(\frac{p}{1-p}\right) = \exp\left(\boldsymbol{w}^\top \boldsymbol{x}\right)$$
$$= e^z$$

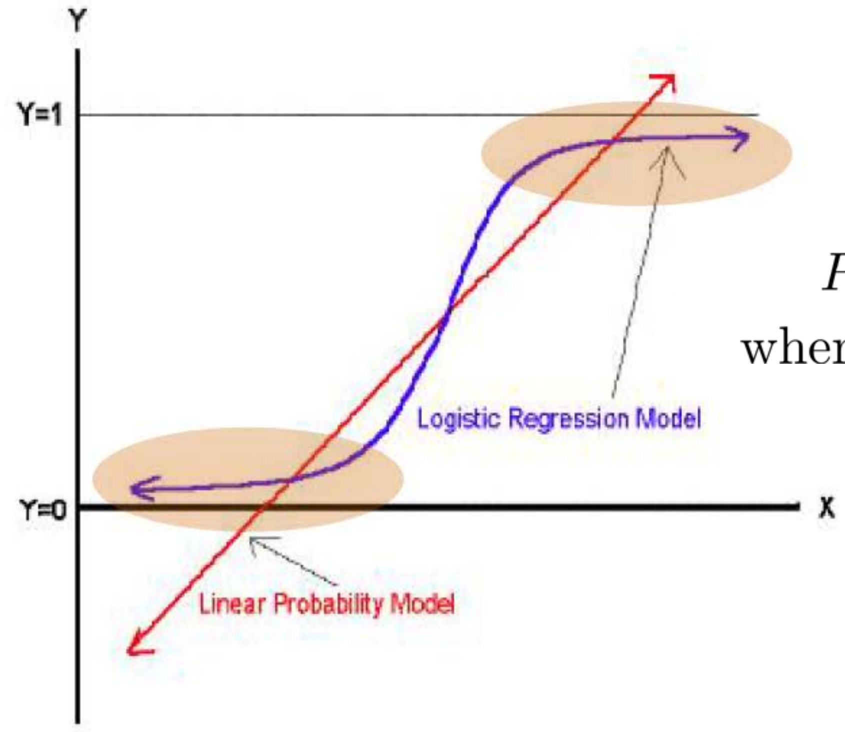
- Recovering ‘p=Pr(Y=1)’

$$p = \frac{\exp\left(\boldsymbol{w}^\top \boldsymbol{x}\right)}{1+\exp\left(\boldsymbol{w}^\top \boldsymbol{x}\right)} = \frac{e^z}{1+e^z} = \boxed{\frac{1}{1+e^{-z}}}$$

Sigmoid function $\sigma(z) : \mathbb{R} \Rightarrow [0,1]$

$$1-p = \frac{1}{1+\exp\left(\boldsymbol{w}^\top \boldsymbol{x}\right)} = \frac{1}{1+e^z} = \frac{e^{-z}}{1+e^{-z}}$$

Logistic regression vs. Linear regression



$$P(Y = 1|X = \boldsymbol{x}) = \frac{1}{1+e^{-z}}$$
where $z = \boldsymbol{w}^\top \boldsymbol{x}$

Log Likelihood

$$Pr(y|x;p) = \begin{cases} p, & y = 1 \\ 1-p, & y = 0 \end{cases}$$
$$= p^y (1-p)^{(1-y)}$$

$$\begin{aligned}
\ell(\boldsymbol{w}) &= \sum_{i=1}^{N} y_i \log\ p(\boldsymbol{x}_i;\boldsymbol{w}) + (1-y_i)\log\left(1 - p(\boldsymbol{x}_i;\boldsymbol{w})\right) \\
&= \sum_{i=1}^{N}\ y_i \log \frac{p(\boldsymbol{x}_i;\boldsymbol{w})}{1 - p(\boldsymbol{x}_i;\boldsymbol{w})} + \log\left(\frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right) \\
&= \sum_{i=1}^{N}\ y_i \boldsymbol{w}^\top \boldsymbol{x}_i - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_i))
\end{aligned}$$

Maximum Likelihood Estimation

$$\frac{\partial}{\partial w^j}\ell(\boldsymbol{w}) = \frac{\partial}{\partial w^j}\left(\sum_{i=1}^{N} y_i \boldsymbol{w}^\top \boldsymbol{x}_i - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}))\right)$$

$$= \sum_{i=1}^{N} x_i^j \underbrace{(y_i - p(\boldsymbol{x}_i;\boldsymbol{w}))}_{\text{Prediction error in probability}}$$

Gradient ascent

$$w^j(t+1) = w^j(t) + \eta \sum_{i=1}^{N} x_i^j \left(y_i - p(\boldsymbol{x}_i; \boldsymbol{w})\right)$$

- Iteratively updating the weights in this fashion increases likelihood each round.
- We eventually reach the maximum
- We are near the maximum when changes in the weights are small.
- Thus, we can stop when the sum of the absolute values of the weight differences is less than some small number.

Summary

- Logistic regression gives us a framework in which to model binary outcomes
- Uses the structure of linear models, with outcomes (log(odds)) modelled as a function of input variables
- Many concepts carry over from linear regression
  - logistic regression
  - Kernelization (using non-linear functions of input variables)